第3章 基于梯度下降法的RBF神经网络控制

离散神经网络控制系统中，常采用梯度下降法实现神经网络权值的学习，有代表性的研究工作如文献[1,2]。

3.1 基于RBF神经网络的监督控制

3.1.1 RBF监督控制

图3.1为基于RBF神经网络的监督控制系统，其控制思想为：初始阶段采用PD反馈控制，然后过渡到神经网络控制。在控制过程中，如出现较大的误差，则PD控制起主导作用，神经网络控制起调节作用。

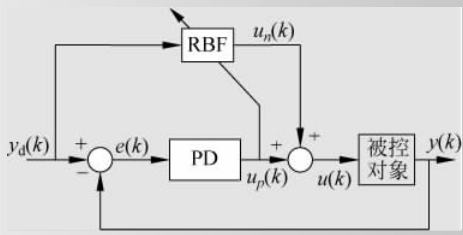

图3.1 基于RBF神经网络的监督控制系统

设径向基向量为 $\boldsymbol{h}=[h_1,\cdots,h_m]^{\mathrm{T}}$，$h_j$ 为高斯函数，则

$$h_j=\exp\left(-\frac{\|\boldsymbol{x}(k)-\boldsymbol{c}_j\|^2}{2b_j^2}\right) \tag{3.1}$$

其中，$i=1,j=1,\cdots,m$；$\boldsymbol{x}(k)$ 为RBF网络的输入；$\boldsymbol{c}_j=[c_{11},\cdots,c_{1m}]$；$\boldsymbol{b}=[b_1,\cdots,b_m]^{\mathrm{T}}$。

设权值向量为

$$\boldsymbol{w}=[w_1,\cdots,w_m]^{\mathrm{T}} \tag{3.2}$$

RBF神经网络的输出为

$$u_n(k)=h_1w_1+\cdots+h_jw_j+\cdots+h_mw_m \tag{3.3}$$

其中，m 为隐含层节点的个数。

总控制输入为 $u(k)=u_n(k)+u_p(k)$，误差指标为

$$E(k)=\frac{1}{2}(u_n(k)-u(k))^2 \tag{3.4}$$

采用梯度下降法，网络权值学习算法为

$$\Delta w_j(k)=-\eta\frac{\partial E(k)}{\partial w_j(k)}=\eta(u_n(k)-u(k))h_j(k)$$

$$\boldsymbol{w}(k)=\boldsymbol{w}(k-1)+\Delta\boldsymbol{w}(k)+\alpha(\boldsymbol{w}(k-1)-\boldsymbol{w}(k-2)) \tag{3.5}$$

其中，$\eta\in[0,1]$为学习速率，$\alpha\in[0,1]$为动量因子。

3.1.2 仿真实例

设控制对象为

$$G(s)=\frac{1000}{s^3+87.35s^2+10470s}$$

取采样周期为1ms，对上述对象进行离散化，可得

$$y(k)=-\mathrm{den}(2)y(k-1)-\mathrm{den}(3)y(k-2)+\mathrm{num}(2)u(k-1)+\mathrm{num}(3)u(k-2)$$

取神经网络的结构为1-4-1，理想跟踪指令为$\boldsymbol{x}(k)=y_d(k)$，网络的初始权值取[0,1]之间的随机数，根据网络的输入范围，高斯函数参数取$\boldsymbol{c}=[-5\quad -3\quad 0\quad 3\quad 5]^T$。取学习速率$\eta=0.30$，动量因子$\alpha=0.05$。

仿真结果如图3.2和图3.3所示。RBF监督控制的仿真程序为chap3_1.m，详见附录。

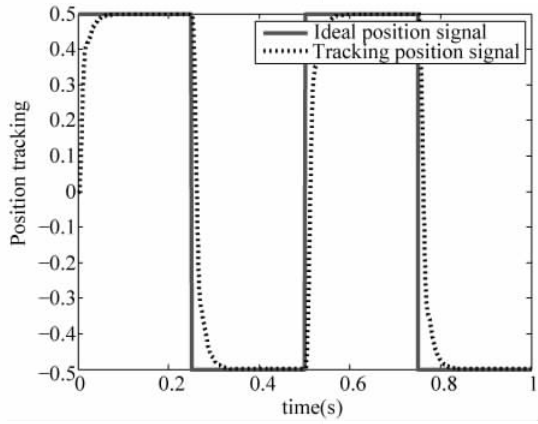

图3.2 方波跟踪效果

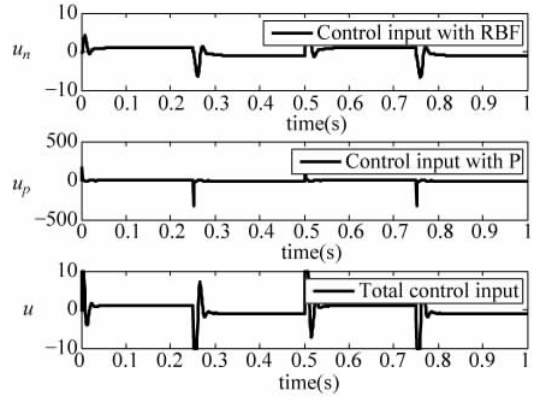

图3.3 神经网络输入、PD控制输入及总控制输入

3.2 基于 RBF 神经网络的模型参考自适应

3.2.1 控制系统设计

图 3.4 为基于 RBF 神经网络的模型参考自适应控制系统框图。

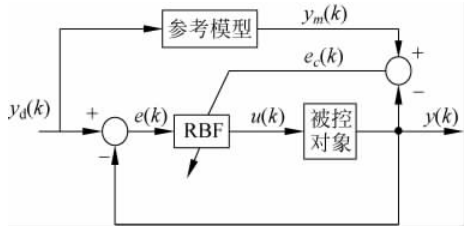

图 3.4 基于 RBF 神经网络的模型参考自适应控制系统

设理想跟踪指令为 $y_m(k)$，则定义跟踪误差为

$$e(k)=y_m(k)-y(k) \tag{3.6}$$

网络权值学习误差指标为

$$E(k)=\frac{1}{2}e(k)^2 \tag{3.7}$$

控制输入为 RBF 网络的输出：

$$u(k)=h_1w_1+\cdots+h_jw_j+\cdots+h_mw_m \tag{3.8}$$

其中，m 为隐含层的节点个数，w_j 为节点的权值，h_j 为高斯基函数的输出。

在 RBF 网络中，$\boldsymbol{x}=[x_1,\cdots,x_n]^{\mathrm{T}}$ 为网络输入，$\boldsymbol{h}=[h_1,\cdots,h_m]^{\mathrm{T}}$，$h_j$ 为高斯函数：

$$h_j=\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|^2}{2b_j^2}\right) \tag{3.9}$$

其中，$i=1,\cdots,n$，$j=1,\cdots,m$。$b_j>0$，$\boldsymbol{c}_j=[c_{j1},\cdots,c_{ji},\cdots,c_{jn}]$，$\boldsymbol{b}=[b_1,\cdots,b_m]^{\mathrm{T}}$。

设权值向量为

$$\boldsymbol{w}=[w_1,\cdots,w_m]^{\mathrm{T}} \tag{3.10}$$

由梯度下降法，网络的学习算法为

$$\Delta w_j(k)=-\eta\frac{\partial E(k)}{\partial w}=\eta e_c(k)\frac{\partial y(k)}{\partial u(k)}h_j$$

$$w_j(k)=w_j(k-1)+\Delta w_j(k)+\alpha\Delta w_j(k) \tag{3.11}$$

其中，η 为学习速率；α 为动量因子；$\eta\in[0,1]$，$\alpha\in[0,1]$。

同理可得

$$\Delta b_j(k)=-\eta\frac{\partial E(k)}{\partial b_j}=\eta e_c(k)\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial b_j}$$

$$=\eta e_c(k)\frac{\partial y(k)}{\partial u(k)}w_jh_j\frac{\|\boldsymbol{x}-\boldsymbol{c}_{ij}\|^2}{b_j^3} \tag{3.12}$$

$$b_j(k)=b_j(k-1)+\eta\Delta b_j(k)+\alpha(b_j(k-1)-b_j(k-2)) \tag{3.13}$$

$$\Delta c_{ij}(k)=-\eta\frac{\partial E(k)}{\partial c_{ij}}=\eta e_c(k)\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial c_{ij}}$$

$$= \eta e_c(k)\frac{\partial y(k)}{\partial u(k)} w_j h_j \frac{x_i - c_{ij}}{b_j^2} \tag{3.14}$$

$$c_{ij}(k) = c_{ij}(k-1) + \eta \Delta c_{ij}(k) + \alpha(c_{ij}(k-1) - c_{ij}(k-2)) \tag{3.15}$$

其中，$\frac{\partial y(k)}{\partial u(k)}$为 Jacobian 阵，表征系统输出对控制输入的灵敏度。

3.2.2 仿真实例

取离散被控对象为

$$y(k) = (-0.10y(k-1) + u(k-1))/(1 + y(k-1)^2)$$

其中，采样周期为 $ts=1$ms，参考模型为 $y_m(k)=0.6y_m(k-1)+y_d(k)$，理想跟踪指令为 $y_d(k)=0.50\sin(2\pi k \times ts)$。

取 RBF 神经网络的输入为 $y_d(k)$、$e(k)$和 $u(k)$，学习速率为 $\eta=0.35$，动量因子为$\alpha=0.05$。

高斯函数参数初值为 $\boldsymbol{c}=\begin{bmatrix} -3 & -2 & -1 & 1 & 2 & 3 \\ -3 & -2 & -1 & 1 & 2 & 3 \\ -3 & -2 & -1 & 1 & 2 & 3 \end{bmatrix}^{\mathrm{T}}$，$\boldsymbol{b}=[2,2,2,2,2,2]^{\mathrm{T}}$，网络初始权值为 $\boldsymbol{w}=[-0.0316 \quad -0.0421 \quad -0.0318 \quad 0.0068 \quad 0.0454 \quad -0.0381]$。

仿真结果如图 3.5 和图 3.6 所示。基于 RBF 神经网络的模型参考自适应控制程序为 chap3_2.m。

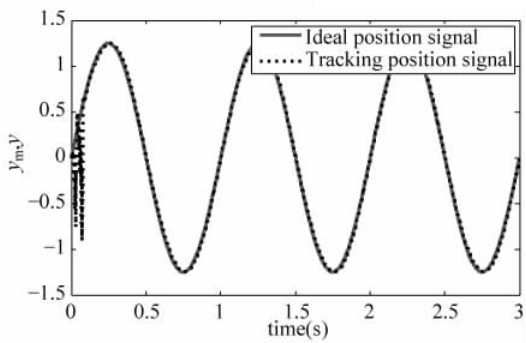

图 3.5 正弦跟踪

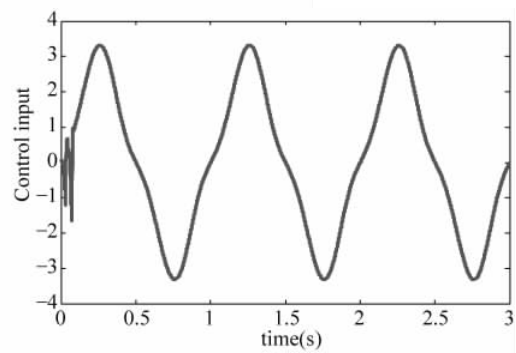

图 3.6 控制输入

3.3 RBF自校正控制

3.3.1 系统描述

考虑被控对象为

$$y(k+1)=g[y(k)]+\phi[y(k)]u(k) \tag{3.16}$$

其中，$y(k)$为系统输出；$u(k)$控制输入。

设 $y_d(k)$为理想跟踪指令。如果 $g[\cdot]$和 $\phi[\cdot]$为已知的，设计自校正控制器为

$$u(k)=\frac{-g[\cdot]}{\phi[\cdot]}+\frac{y_d(k+1)}{\phi[\cdot]} \tag{3.17}$$

在实际工程中 $g[\cdot]$和 $\phi[\cdot]$通常是未知的，控制器式(3.17)难以实现，采用RBF神经网络逼近 $g[\cdot]$和 $\phi[\cdot]$，可有效解决这一难题。

3.3.2 RBF控制算法设计

如果 $g[\cdot]$和 $\phi[\cdot]$未知，可利用RBF网络逼近 $g[\cdot]$和$\phi[\cdot]$，从而得到 $g[\cdot]$和$\phi[\cdot]$的估计值，记为 $Ng[\cdot]$和$N\phi[\cdot]$，则自校正控制器为

$$u(k)=\frac{-Ng[\cdot]}{N\phi[\cdot]}+\frac{y_d(k+1)}{N\phi[\cdot]} \tag{3.18}$$

其中，$Ng[\cdot]$和 $N\phi[\cdot]$为RBF神经网络的逼近输出。

分别采用两个RBF神经网络逼近 $g[\cdot]$和 $\phi[\cdot]$，$\boldsymbol{W}$ 和 $\boldsymbol{V}$ 分别为两个神经网络的权值。在RBF网络设计中，取 $y(k)$为网络输入，$\boldsymbol{h}=[h_1,\cdots,h_m]^{\mathrm{T}}$，$h_j$ 为高斯函数：

$$h_j=\exp\left(-\frac{\|y(k)-\boldsymbol{c}_j\|^2}{2b_j^2}\right) \tag{3.19}$$

其中，$i=1$；$j=1,\cdots,m$；$b_j>0$；$\boldsymbol{c}_j=[c_{11},\cdots,c_{1m}]$；$\boldsymbol{b}=[b_1,\cdots,b_m]^{\mathrm{T}}$。

设RBF网络的权值为

$$\boldsymbol{W}=[w_1,\cdots,w_m]^{\mathrm{T}} \tag{3.20}$$

$$\boldsymbol{V}=[v_1,\cdots,v_m]^{\mathrm{T}} \tag{3.21}$$

两个RBF网络的输出分别为

$$Ng(k)=h_1w_1+\cdots+h_jw_j+\cdots+h_mw_m \tag{3.22}$$

$$N\varphi(k)=h_1v_1+\cdots+h_jv_j+\cdots+h_mv_m \tag{3.23}$$

其中，m 为隐含层节点的个数。

基于RBF神经网络逼近的输出为

$$y_m(k)=Ng[y(k-1);W(k)]+N\phi[y(k-1);V(k)]u(k-1) \tag{3.24}$$

基于神经网络 $Ng[\cdot]$和 $N\phi[\cdot]$逼近的自适应控制系统框图如图3.7所示。

用于权值调整的误差指标为

$$E(k)=\frac{1}{2}(y(k)-y_m(k))^2 \tag{3.25}$$

根据梯度下降法，网络权值学习算法为

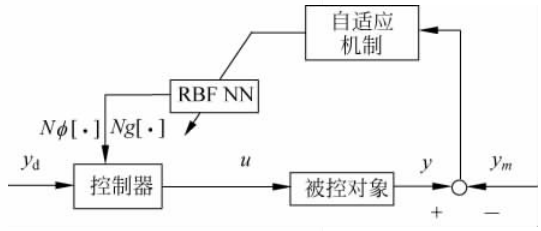

图 3.7　基于 RBF 逼近的自适应控制系统

$$\Delta w_j(k) = -\eta_w \frac{\partial E(k)}{\partial w_j(k)} = \eta_w (y(k) - y_m(k)) h_j(k)$$

$$\Delta v_j(k) = -\eta_v \frac{\partial E(k)}{\partial v_j(k)}$$

$$= \eta_v (y(k) - y_m(k)) h_j(k) u(k-1)$$

$$\boldsymbol{W}(k) = \boldsymbol{W}(k-1) + \Delta \boldsymbol{W}(k) + \alpha(\boldsymbol{W}(k-1) - \boldsymbol{W}(k-2)) \tag{3.26}$$

$$\boldsymbol{V}(k) = \boldsymbol{V}(k-1) + \Delta \boldsymbol{V}(k) + \alpha(\boldsymbol{V}(k-1) - \boldsymbol{V}(k-2)) \tag{3.27}$$

其中，η_w 和 η_w 为学习速率，α 为动量因子。

3.3.3　仿真实例

取被控对象为

$$y(k) = 0.8\sin(y(k-1)) + 15u(k-1)$$

其中，$g[y(k)] = 0.8\sin(y(k-1))$，$\phi[y(k)] = 15$。

理想跟踪指令为 $y_d(t) = 2.0\sin(0.1\pi t)$，RBF 神经网络的结构为 1-6-1，网络的初始权值和高斯函数参数分别设置为 $\boldsymbol{W} = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]^T$，$\boldsymbol{V} = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]^T$，$\boldsymbol{c}_j = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]^T$，$\boldsymbol{b} = [5, 5, 5, 5, 5, 5]^T$。

取学习速率为 $\eta_1 = 0.15$，$\eta_2 = 0.50$，动量因子为 $\alpha = 0.05$。仿真结果如图 3.8～图 3.10 所示。RBF 神经网络自校正控制程序为 chap3_3.m，详见附录。

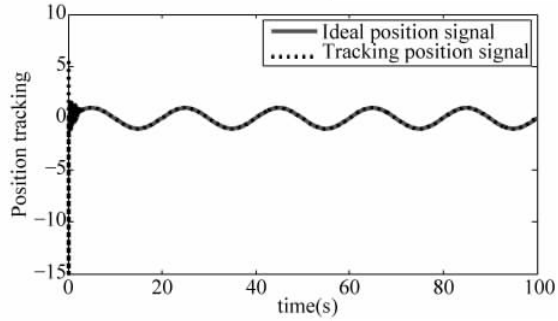

图 3.8　正弦指令的跟踪

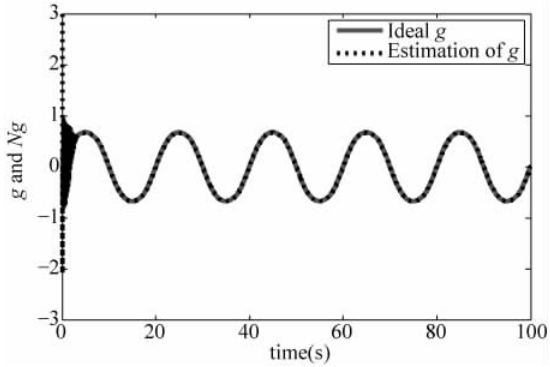

图 3.9 $g(x,t)$及其估计值$\hat{g}(x,t)$

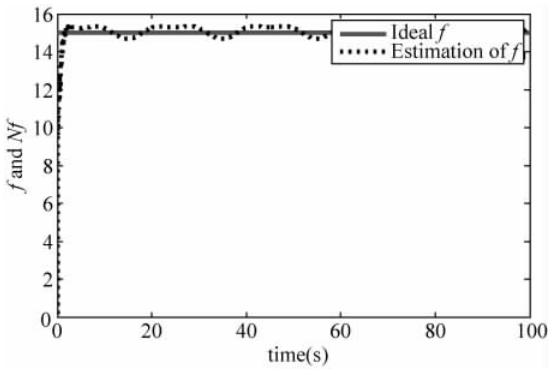

图 3.10 $f(x,t)$及其估计值$\hat{f}(x,t)$

梯度下降法的优点是设计过程简单,但收敛效果取决于初值的选取,采用该方法调整神经网络权值易陷入局部最优,不能保证闭环系统的全局稳定性,因此有很大的局限性。

附录 仿真程序

3.1.2 节的程序

RBF 监督控制仿真程序:chap3_1.m

```
% RBF Supervisory Control
clear all;
close all;

ts = 0.001;
sys = tf(1000,[1,50,2000]);
dsys = c2d(sys,ts,'z');
[num,den] = tfdata(dsys,'v');

y_1 = 0;y_2 = 0;
u_1 = 0;u_2 = 0;
```

```
e_1 = 0;

xi = 0;
x = [0,0]';

b = 0.5 * ones(4,1);
c = [ -2  -1 1 2];
w = rands(4,1);
w_1 = w;
w_2 = w_1;

xite = 0.30;
alfa = 0.05;

kp = 25;
kd = 0.3;
for k = 1:1:1000
    time(k) = k * ts;
S = 1;
if S == 1
   yd(k) = 0.5 * sign(sin(2 * 2 * pi * k * ts));    % Square Signal
elseif S == 2
   yd(k) = 0.5 * (sin(3 * 2 * pi * k * ts));        % Square Signal
end

y(k) = -den(2) * y_1 - den(3) * y_2 + num(2) * u_1 + num(3) * u_2;
e(k) = yd(k) - y(k);

xi = yd(k);

for j = 1:1:4
    h(j) = exp( -norm(xi - c(:,j))^2/(2 * b(j) * b(j)));
end
un(k) = w' * h';

 % PD Controller
up(k) = kp * x(1) + kd * x(2);

M = 2;
if M == 1                  % Only Using PID Control
   u(k) = up(k);
elseif M == 2              % Total control output
    u(k) = up(k) + un(k);
end

if u(k)>=10
   u(k) = 10;
end
if u(k)<= -10
   u(k) = -10;
end
```

```
% Update NN Weight
d_w = -xite*(un(k)-u(k))*h';
w = w_1 + d_w + alfa*(w_1 - w_2);

w_2 = w_1;
w_1 = w;
u_2 = u_1;
u_1 = u(k);
y_2 = y_1;
y_1 = y(k);

x(1) = e(k);              % Calculating P
x(2) = (e(k) - e_1)/ts;   % Calculating D
e_1 = e(k);
end
figure(1);
plot(time,yd,'r',time,y,'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('Ideal position signal','Tracking position signal');

figure(2);
subplot(311);
plot(time,un,'k','linewidth',2);
xlabel('time(s)');ylabel('un');
legend('Control input with RBF');
subplot(312);
plot(time,up,'k','linewidth',2);
xlabel('time(s)');ylabel('up');
legend('Control input with P');
subplot(313);
plot(time,u,'k','linewidth',2);
xlabel('time(s)');ylabel('u');
legend('Total control input');
```

3.2.2 节的程序

基于 RBF 神经网络的模型参考自适应控制：chap3_2.m

```
% Model Reference Adaptive RBF Control
clear all;
close all;

u_1 = 0;
y_1 = 0;
ym_1 = 0;

x = [0,0,0]';
c = [-3 -2 -1 0 1 2 3;
     -3 -2 -1 0 1 2 3;
     -3 -2 -1 0 1 2 3];
```

```
b = 2;
w = rands(1,7);

xite = 0.35;
alfa = 0.05;
h = [0,0,0,0,0,0,0]';

c_1 = c;c_2 = c;
b_1 = b;b_2 = b;
w_1 = w;w_2 = w;

ts = 0.001;
for k = 1:1:3000
time(k) = k * ts;

yd(k) = 0.50 * sin(2 * pi * k * ts);
ym(k) = 0.6 * ym_1 + yd(k);

y(k) = ( - 0.1 * y_1 + u_1)/(1 + y_1 ^2); % Nonlinear plant

for j = 1:1:7
    h(j) = exp( - norm(x - c(:,j))^2/(2 * b ^2));
end
u(k) = w * h;

ec(k) = ym(k) - y(k);
dyu(k) = sign((y(k) - y_1)/(u(k) - u_1));

d_w = 0 * w;
for j = 1:1:7
    d_w(j) = xite * ec(k) * h(j) * dyu(k);
end
w = w_1 + d_w + alfa * (w_1 - w_2);
 % Return of parameters
u_1 = u(k);
y_1 = y(k);
ym_1 = ym(k);

x(1) = yd(k);
x(2) = ec(k);
x(3) = y(k);

w_2 = w_1;w_1 = w;
end
figure(1);
plot(time,ym,'r',time,y,'k:','linewidth',2);
xlabel('time(s)');ylabel('ym,y');
legend('Ideal position signal','Tracking position signal');
figure(2);
plot(time,u,'r','linewidth',2);
xlabel('time(s)');ylabel('Control input');
```

3.3.3 节的程序

RBF 神经网络自校正控制：chap3_3.m

```
% Self - Correct control based RBF Identification
clear all;
close all;

xite1 = 0.15;
xite2 = 0.50;
alfa = 0.05;

w = 0.5 * ones(6,1);
v = 0.5 * ones(6,1);

cij = 0.50 * ones(1,6);
bj = 5 * ones(6,1);
h = zeros(6,1);

w_1 = w;w_2 = w_1;
v_1 = v;v_2 = v_1;
u_1 = 0;y_1 = 0;

ts = 0.02;
for k = 1:1:5000
time(k) = k * ts;
yd(k) = 1.0 * sin(0.1 * pi * k * ts);

% Practical Plant;
g(k) = 0.8 * sin(y_1);
f(k) = 15;
y(k) = g(k) + f(k) * u_1;

for j = 1:1:6
    h(j) = exp( - norm(y(k) - cij(:,j))^2/(2 * bj(j) * bj(j)));
end

Ng(k) = w' * h;
Nf(k) = v' * h;

ym(k) = Ng(k) + Nf(k) * u_1;

e(k) = y(k) - ym(k);

d_w = 0 * w;
for j = 1:1:6
    d_w(j) = xite1 * e(k) * h(j);
end
w = w_1 + d_w + alfa * (w_1 - w_2);
```

```
d_v = 0 * v;
for j = 1:1:6
    d_v(j) = xite2 * e(k) * h(j) * u_1;
end
v = v_1 + d_v + alfa * (v_1 - v_2);

u(k) = -Ng(k)/Nf(k) + yd(k)/Nf(k);

u_1 = u(k);
y_1 = y(k);

w_2 = w_1;
w_1 = w;

v_2 = v_1;
v_1 = v;
end
figure(1);
plot(time,yd,'r',time,y,'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('Ideal position signal','Tracking position signal');
figure(2);
plot(time,g,'r',time,Ng,'k:','linewidth',2);
xlabel('time(s)');ylabel('g and Ng');
legend('Ideal g','Estimation of g');
figure(3);
plot(time,f,'r',time,Nf,'k:','linewidth',2);
xlabel('time(s)');ylabel('f and Nf');
legend('Ideal f','Estimation of f');
```

参考文献

[1] Noriega J R, Wang H (1998). A direct adaptive neural network control for unknown nonlinear systems and its application. IEEE Trans Neural Netw 9(1):27～34.

[2] Suykens J A K, Vandewalle J P L, Demoor B L R (1996). Artificial neural networks for modeling and control of nonlinear systems. Kluwer Academic, Boston.

[3] 刘金琨. 智能控制. 2 版. 北京：电子工业出版社,2012.